Technical Report 1548
August 1992

Environment/Tool Integrator for Software Development

Version 1.1

N. T. Tran
R. H. Mumm

Approved for public release; distribution is unlimited.

Technical Report 1548
August 1992

Environment/Tool Integrator for Software Development

Version 1.1

N. T. Tran
R. H. Mumm

**NAVAL COMMAND, CONTROL AND
OCEAN SURVEILLANCE CENTER
RDT&E DIVISION
San Diego, California 92152-5000**

J. D. FONTANA, CAPT, USN
Commanding Officer

R. T. SHEARER
Executive Director

ADMINISTRATIVE INFORMATION

This work was done under the Computer Technology block program 0602234N, Software Engineering for Command, Control, Communications Systems project ECB3, and Software Engineering Environment Prototypes task DN088690.

Released by
G. Schulte, Head
Computer Software and
Technology Branch

Under authority of
A. G. Justice, Head
Information Processing
and Displaying Division

ACKNOWLEDGMENTS

The authors thank Sharon Keith, Sharon Rotter, and Barry Siegel of NRaD for reviewing this document and providing valuable comments.

A special thanks to Dr. Cathy Fernandes of NRaD for her many human factors suggestions. We appreciate all the suggestions from Ron Crepeau of NRaD and from Kent Allen, Tom Van Belle, and others at Intermetrics, Inc.

SM

EXECUTIVE SUMMARY

OBJECTIVE

The objective of this research was to develop a windowing framework for accessing the diverse collections of software tools used by software development projects. A major research goal was to develop an environment/tool integrator (ETI) that can easily be customized to satisfy the unique needs of specific projects.

RESULTS

A prototype version of the ETI was successfully developed and demonstrated. The ETI is being used by the Operations Support System (OSS) project at NRaD. Potential customers in private industry were identified as a result of demonstrations at the two 1992 national Armed Forces Communications and Electronics Association (AFCEA) conferences. The ETI is being submitted to the Software Technology for Adaptable Reliable Systems (STARS) Asset Source for Software Engineering Technology (ASSET) library.

RECOMMENDATIONS

1. More help from human factors experts is needed during the development of user interfaces for DoD software and computer-aided software engineering (CASE) tools. The operation of these user interfaces can and must be simplified.

2. Ada repositories should be cleaned up with help from users. Tools that do not work or are of marginal quality should be removed.

3. The Ada Joint Program Office, National Aeronautics and Space Administration or other government organizations should provide support to AdaNet to improve the quality of tools in the repository. X front-ends, modifications to increase portability, and other enhancements would benefit the entire Ada community.

4. A number of enhancements should be made to the ETI. The customer base should be broadened by porting it to additional Unix-based platforms. A context-sensitive help feature and tool search capability should be provided. Additional Ada, C, and C++ public domain tools are needed. X/Motif front-ends for these tools should be developed to facilitate their use.

CONTENTS

APPENDICES:

FIGURES

ACRONYMS AND ABBREVIATIONS

ABOM	-	Ada Bit-Oriented Message Handler project
ALS/N	-	Ada Language System/Navy
AFCEA	-	Armed Forces Communications and Electronics Association
APRICOT	-	Ada Primitive Compilation Order Tool
ASR	-	Ada Software Repository
ASSET	-	Asset Source for Software Engineering Technology
CARDS	-	Central Archive for Reusable Defense Software
CASE	-	computer-aided software engineering
CIM	-	corporate information management
CRSS	-	C^{3}I Reusable Software System
C^3	-	command control and communications
DoD	-	Department of Defense
DTC-2	-	desk-top computer-2
ETI	-	Environment/Tool Integrator
GUI	-	graphical user interface
IDE	-	Interactive Development Environments
LAN	-	local area network
MACA	-	Management Assistance Corporation of America
MIT	-	Massachusetts Institute of Technology
NASA	-	National Aeronautics and Space Administration
NED	-	Navy Command and Control System Ashore Editor
NRaD	-	Naval Command, Control and Ocean Surveillance Center, Research, Development, Test and Evaluation Division
OSS	-	Operations Support System
SAIC	-	Science Applications International Corporation

SAINT	-	Shared Adaptive Internetworking project
STARS	-	Software Technology for Adaptable Reliable Systems
SWEEP	-	Software Engineering Environment Prototypes
TAC–3	-	Tactical Advanced Computer 3
TI	-	Texas Instruments

1.0 INTRODUCTION

This report describes the environment/tool integrator (ETI) that was developed under the Software Engineering Environment Prototypes (SWEEP) task of the Software Engineering for Command Control and Communications (C^3) Systems project. The ETI is a windowing framework for accessing software tools. It can be easily customized to satisfy the needs of a project. The ETI allows users to define (1) the tool categories that are appropriate for a specific project and (2) the interfaces for accessing individual tools. Tools that may be accessed include computer-aided software engineering (CASE) tools, software engineering environments, reuse libraries, project-specific tools, public domain tools, and utility tools (such as mailer and calculator).

The ETI provides a convenient way of displaying all tools that are available to a project development team. It supports multiple programming languages as well as different applications (e.g., C^3 tactical applications and corporate information management (CIM) applications).

The ETI and all tools that are invoked from it must execute on Unix (SunOS 4.1.1). Tools may reside on various workstations in a local area network (LAN).

The ETI, version 1.1, is being submitted to the Software Technology for Adaptable Reliable Systems (STARS) Asset Source for Software Engineering Technology (ASSET) library, where it will be available to STARS affiliates at no cost. A number of Ada and C public domain tools are included, such as an Ada pretty printer, an Ada line counter, an Ada body stubber, a C pretty printer, a C line counter, and an editor.

The ETI was developed in-house by the Naval Command, Control and Ocean Surveillance Center, Research, Development, Test and Evaluation Division (NRaD). It is currently being used by the Operations Support System (OSS) project at NRaD.

This report includes the following information on Version 1.1 of the ETI:

- Background
- Characteristics
- User instructions
- Lessons learned
- Recommendations
- Description of the tools included

Potential users of the ETI and this report are software developers who use Unix-based computers.

2.0 BACKGROUND

The need for the ETI was recognized by observing the way in which software was being developed for several projects at NRaD and at other Navy laboratories. Problems recognized that can be solved or minimized by using the ETI include the four summarized below.

1. Project software engineers are sometimes unaware of potentially useful tools available for software development, even when they reside on project computers. Very useful tools that can be used are not being applied.

2. Most commercially available environments have extensibility limitations. Examples of limitations include the following:

   - Users cannot add their own tools because the environment is predetermined and cannot be modified by the user.
   - Tool integration is possible but difficult and time consuming.
   - Tool integration is possible but can only be done in such a way that the newly integrated tool is accessed by traversing through layers of menus.
   - Upon the release of a new version of an environment, users must reintegrate project tools.

3. Commercial environments are frequently tied to one computer language or one compiler (theirs). NRaD and Navy projects often use multiple programming languages.

4. Licensing and cost considerations preclude the widespread use of commercial environments.

The ETI addresses each of these four problems.

1. It provides a way in which project users can view and access all available project tools. This is especially helpful for programmers who are new to a project.

2. It provides an alternative to integrating tools into commercial environments. Tools may be integrated into and accessed from the ETI. Multiple commercial environments and reuse libraries can be accessed from it.

3. The ETI is not tied to any one language or compiler. Multiple compilation systems may be accessed.

4. The ETI is public domain software. It is being submitted to the STARS ASSET library.

3.0 CHARACTERISTICS

This section describes the key features of the ETI and the required hardware and software.

3.1 KEY FEATURES

- Provides a capability for using software tools, environments, and reuse libraries in an integrated manner for software development.
- Offers easy customization by users. Project members can define tool categories and tool interfaces by creating a setup table. Knowledge of graphical user interface (GUI) builder tools and X/Motif is not required.
- Includes enhanced public domain Ada and C software development tools and utility tools.
- Provides interfaces to commercial environments and tools.
- Conforms to Navy command and control user interface specifications (Ref. 1).
- Includes on-line help facility.
- Provides portability across platforms (Unix).

3.2 REQUIRED HARDWARE AND SOFTWARE

- Currently executes on Sun 3, Sun 4, DTC-2, and other computers using SunOS 4.1.1.
- Requires X/Motif (X11R4 libraries and Motif 1.1.1).

4.0 USING THE ENVIRONMENT/TOOL INTEGRATOR

4.1 INSTALLATION

Instructions are provided below for installing the ETI from (1) a tape provided by NRaD or (2) the STARS ASSET library.

It is assumed that the user is running the Unix "C-shell" for the Sun SPARC using SunOS 4.1.1 and that he or she has a basic understanding of the Unix operating system commands. Note, if running another shell (e.g., KORN), the installation procedures will be different.

Before following the steps below, move to the directory where the ETI is to be installed using the Unix command "cd". Choose the directory of your cnoice then follow the appropriate set of instructions.

4.1.1 Installation from NRaD Tape

1. Unload Files from Tape

Insert the ETI tape in the tape drive. Unload the files on the tape using the command given below. This will automatically create the directory "sweep1.1":

```
tar xvf /dev/rst0  ./sweep1.1
```

2. Set Up the Environment Variables

Set up the environment variables in the file ".cshrc". This file must be in your home directory. Place the following commands at the bottom of this file using an editor:

```
setenv SWEEP_HOME  <path>/sweep1.1
set path=($SWEEP_HOME/bin $path)
setenv XAPPLRESDIR $SWEEP_HOME/app-defaults/
```

Exit the file.

Note, <path> in the first command, indicates that you must specify the appropriate complete path name.

Next, type the following to set the environment variables:

```
source ~/.cshrc
```

You may confirm the success of the installation by invoking the ETI and noting that the top-level window is displayed on your screen. The invocation step and a picture of the top-level window are given in Section 4.2.1.

4.1.2 Installation from STARS ASSET Library

1. Extract and Transfer Files to Host Computer

a. Get Account on STARS ASSET Computer

Before installing the ETI from the STARS ASSET library, you must first get an account on the ASSET computer. To do this, call 304-594-1762 to get a user form. Fill it out and return it to the ASSET library system administrator.

b. Retrieve the Environment/Tool Integrator

First connect your host computer to the ASSET computer by typing

```
telnet source.asset.com (192.131.125.10)
```

and logging on using your userid and password.

Then, to access the STARS ASSET repository type

```
repos
```

Do an "Asset Search" and "Search" to scroll through the list of assets. When the ETI is highlighted, do an "Extract". This retrieves the ETI files from the library and copies them to your current directory on the ASSET computer.

If detailed instructions for using the ASSET library are needed, go back to the "repos" level menu, select "User´s Manual", and browse to the appropriate instructions.

c. Transfer the Environment/Tool Integrator Files

Transfer the files using "ftp", Kermit, or other file transfer software from the ASSET computer to your host computer.

2. Uncompress and Restore File

Enter the following commands:

```
uncompress sweep.Z
tar xvf sweep
```

The first command uncompresses the single ETI file "sweep.Z", and replaces it with the file "sweep". The second creates the individual files and automatically creates the directory "sweep1.1".

3. Set Up the Environment Variables

Follow the same steps as given in Section 4.1.1., subparagraph 2 above.

4.2 EXECUTION

4.2.1 Invoking the Environment/Tool Integrator

Execute the ETI (in any directory) by typing

```
sweep
```

The top-level window shown in figure 1 will appear on the user's monitor. This window may be tailored to fit the specific needs of individual projects. Tailoring is discussed in Section 4.3.

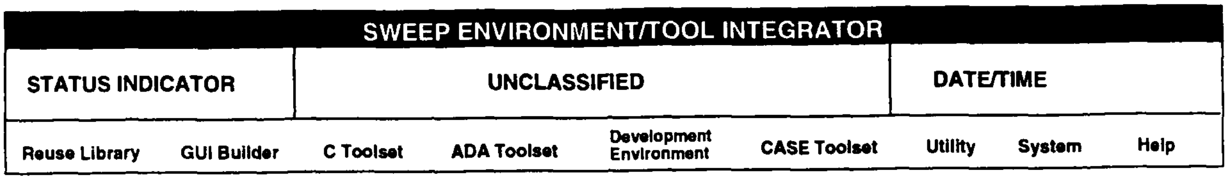


Figure 1. Top-level window.

4.2.2 Top-Level Window Description

At the top of figure 1, below the line "SWEEP ENVIRONMENT/TOOL INTEGRATOR", is the "STATUS INDICATOR" position, where a short message is displayed if a problem occurs. Following this is the "SECURITY CLASSIFICATION" position which, in this example, is set to "UNCLASSIFIED". This position represents the highest level of classification for any project data. The "DATE/TIME" position indicates the current date and time based on the system date and time.

The next line shows the seven tool categories as well as the "System" and "Help" capabilities.

Figure 2 shows the individual tools within each tool category and the options available under "System" and "Help". In this figure, all options are displayed. (This is done for explanatory purposes only. In actual use, tools may be displayed for only one category at a time.)

In figure 2, the tools that are displayed in bold print (black on the monitor) are included with Version 1.1 of the ETI. Those that are not included appear in italic print in the figure (gray on the monitor). Tools displayed in gray have not been installed. Most tools displayed in gray are commercially available and, if needed by the project, must be purchased. Note that figure 2 shows that the current (ETI Version 1.1) C Toolset contains a line counter and pretty printer, that the Ada Toolset contains a body stubber, line counter, pager, and pretty printer, and that the utility tools are a calculator, clip board, command shell, editors, E-mailer, and file manager. The right arrow next to "Editor" indicates that multiple editors are provided.

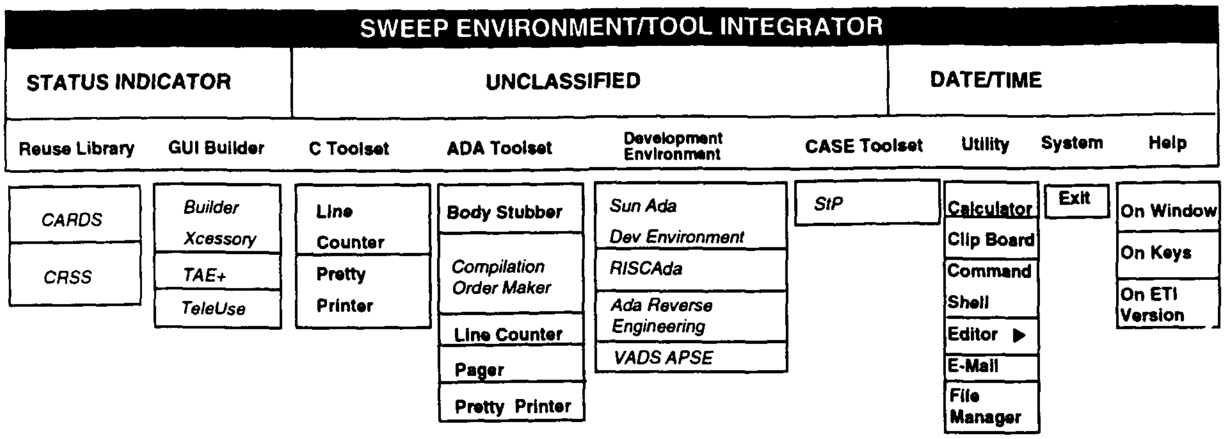


Figure 2. Top-level window with tools and options displayed.

Next, note the options that are provided under "System" and "Help", on the right-hand side of the window. "System" is for executing the ETI system functions. In Version 1.1, the only option available is to exit the ETI. The "Help" capability provides the user with (1) help instructions for the top-level window, (2) help instructions for using the keyboard function keys, and (3) ETI version information.

The ETI is initially configured to include the seven tool categories and the tools indicated in figure 2. This configuration may be changed by following the tailoring instructions given in Section 4.3. The number of tool categories, names of the tool categories, and tools within each may be changed. All projects probably will want to keep the "Utility" and "Help" categories. "System" is the only category that the user cannot customize.

Figure 3 shows the subcategory capability. The right arrow pointing from "Editor" indicates that multiple editors may be selected. The editors within this subcategory are the NED, Emacs, and Xedit editors. A brief description of the NED is contained in Appendix A. A more complete description is contained in Ref. 2. NED complies with the Navy Command and Control user interface specifications (Ref. 1). Emacs is the GNU Emacs editor, which is in the public domain. Xedit is an X editor, which is also in the public domain.

4.2.3 Tool Invocation

Individual tools are invoked in the standard Motif manner: click the left mouse button on the tool category and then click the left mouse button on the appropriate tool. A description of the ETI tools is given in Appendix A.

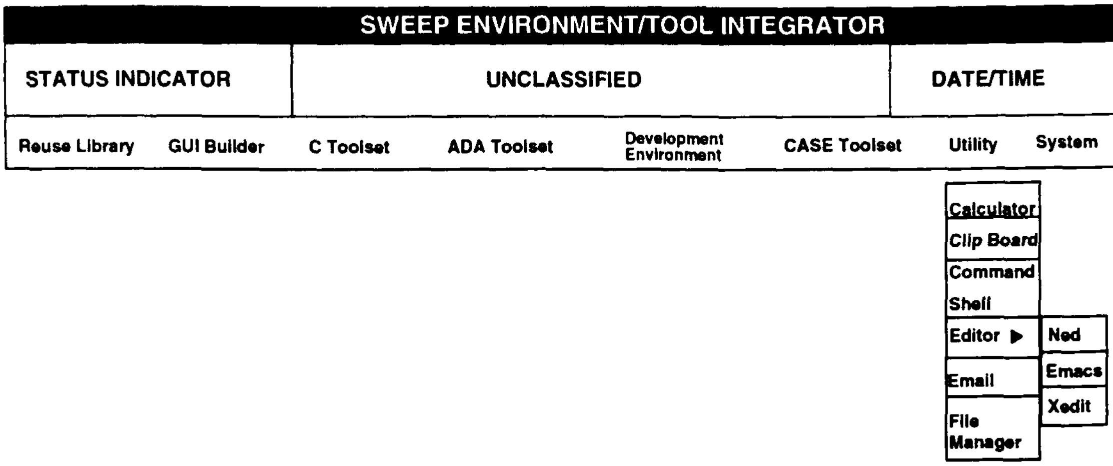


Figure 3. Top-level window with subcategory of editors displayed.

4.3 TAILORING

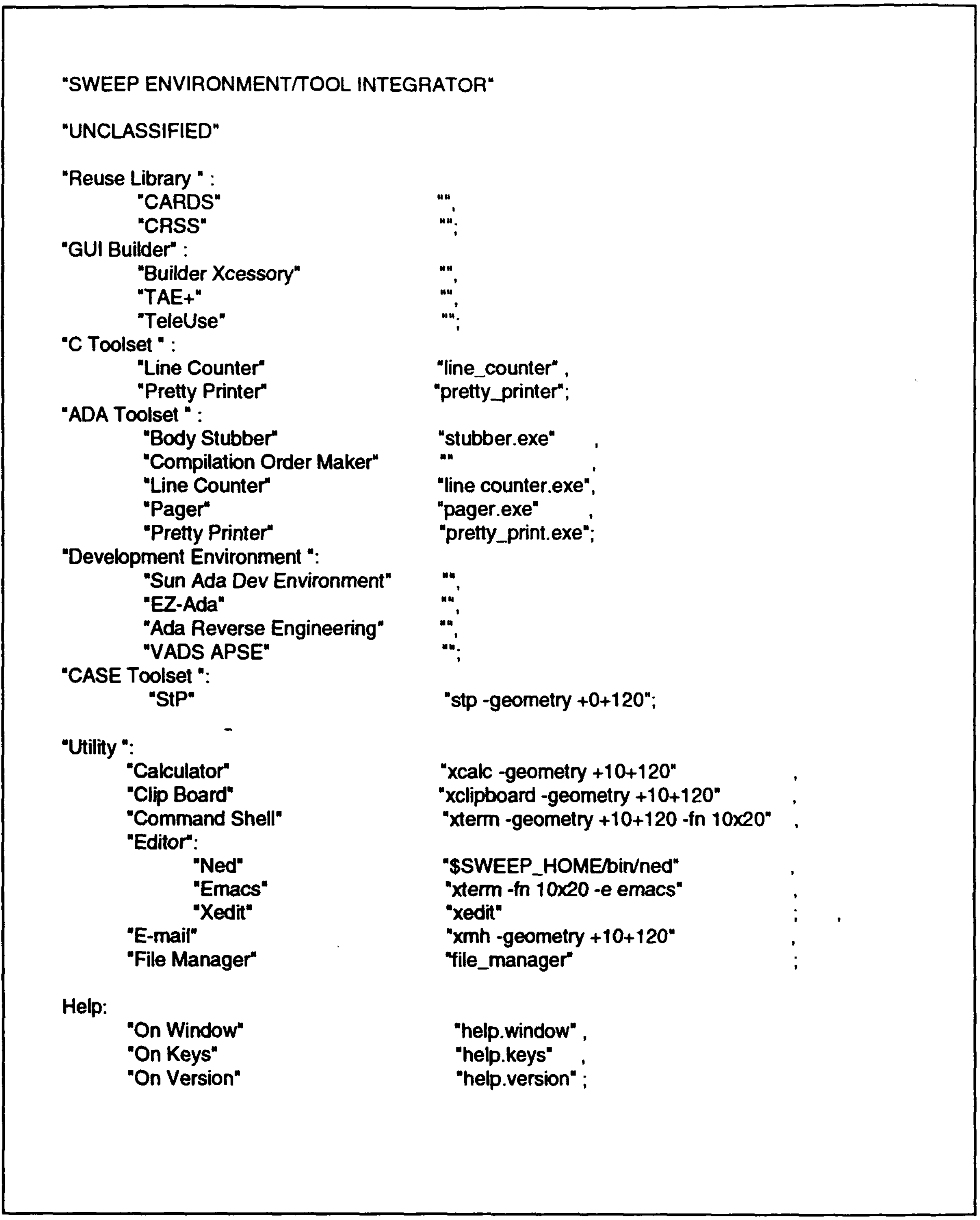
Users are allowed to define the name of the top-level window, the classification of project data, the number of tool categories, the names of the tool categories, and the tools within each category. However, to be in compliance with the Navy Command and Control user interface specifications (Ref. 1), the ETI should not have more than ten categories plus "Help" and should not have more than ten tools within each category.

4.3.1 Version 1.1 as Delivered

The window and tool categories shown in figure 2 are automatically produced from the setup table shown in figure 4. The setup table, contained in the file "sweep_ config", was automatically loaded during installation. The setup table is free format (i.e., there is at least one blank space separating each field).

4.3.1.1 Setup Table. At the top of the setup table, the field "SWEEP ENVIRONMENT/TOOL INTEGRATOR" provides the name of the top-level window, which is displayed at the top of figure 2. The next field in figure 4, "UNCLASSIFIED", gives the highest level of classification of the project data.

Next, note the tool category names and the tool names on the left side of the table. Both must be enclosed in quotes. To the right of each tool name is the executable field. This field can be represented in three different ways:



```
"SWEEP ENVIRONMENT/TOOL INTEGRATOR"

"UNCLASSIFIED"

"Reuse Library " :
        "CARDS"                      "",
        "CRSS"                       "";
"GUI Builder" :
        "Builder Xcessory"           "",
        "TAE+"                       "",
        "TeleUse"                    "";
"C Toolset " :
        "Line Counter"               "line_counter" ,
        "Pretty Printer"             "pretty_printer";
"ADA Toolset " :
        "Body Stubber"               "stubber.exe"     ,
        "Compilation Order Maker"    ""                ,
        "Line Counter"               "line counter.exe",
        "Pager"                      "pager.exe"       ,
        "Pretty Printer"             "pretty_print.exe";
"Development Environment ":
        "Sun Ada Dev Environment"    "",
        "EZ-Ada"                     "",
        "Ada Reverse Engineering"    "",
        "VADS APSE"                  "";
"CASE Toolset ":
        "StP"                        "stp -geometry +0+120";

"Utility ":
      "Calculator"                   "xcalc -geometry +10+120"                ,
      "Clip Board"                   "xclipboard -geometry +10+120"           ,
      "Command Shell"                "xterm -geometry +10+120 -fn 10x20"      ,
      "Editor":
            "Ned"                    "$SWEEP_HOME/bin/ned"                    ,
            "Emacs"                  "xterm -fn 10x20 -e emacs"               ,
            "Xedit"                  "xedit"                                  ;   ,
      "E-mail"                       "xmh -geometry +10+120"                  ,
      "File Manager"                 "file_manager"                           ;

Help:
      "On Window"                     "help.window" ,
      "On Keys"                       "help.keys"   ,
      "On Version"                    "help.version" ;
```

Figure 4. Example of setup table.

1. It can be the path name with the executable file name. This string must be enclosed in quotes.

   For example, look about two thirds of the way down figure 4. In the "Utility" category, "Editor" subcategory, editor "Ned", the path name to the NED executable is the path name including the executable file, "$SWEEP_HOME/bin/ned".

2. It can be the executable file name enclosed in quotes (without the path name). In this case, the path name must be set in the ".cshrc" file.

   For example, in the "C Toolset" category (towards the top of figure 4), for "Line Counter", the executable field is "line_counter", which indicates that the path name is in ".cshrc".

3. It can be a pair of contiguous quotes. This indicates that the executable file name has not been specified. This may be used, for example, to show that there is a place for the tool, but that it has not been purchased or is not presently available.

   The "CARDS" and "CRSS" library executable fields are specified in this manner.

Also note that subcategories of tools can be defined. For example, the "Utility" category has the "Editor" subcategory that consists of the three editors: Ned, Emacs, and Xedit.

Finally, note the file names "help.window", "help.keys", and "help.version", on the bottom right of the table under "Help" category. These files contain text for the on-line help and must be placed in the directory "$SWEEP_HOME/help".

Keep the following rules in mind when creating a setup table.

- Each tool category name is enclosed in quotes followed by a colon (with the exception of the "help" category).
- Each tool name is enclosed in quotes.
- The delimiter separating tool information is a comma.
- The end of the category information is indicated with a semicolon.
- When the name of the executable file is not given within quotes, the tool will appear in gray on the monitor.
- When there is a subcategory of tools, the rules are analogous to those for a category of tools.

- The subcategory name is enclosed in quotes followed by a colon.
- Each tool name is enclosed in quotes.
- The delimiter separating tool information is a comma.
- The end of the subcategory is indicated with a semicolon.

4.3.1.2. X Instructions. Note in figure 4 that sometimes following the executable file name, there are additional instructions. These are command line X instructions that the user may or may not need.

A short explanation of these instructions is provided here. A detailed explanation is found in the on-line user's manual "man pages" (Ref. 3), which is included with the X Window System from the Massachusetts Institute of Technology (MIT).

Find the executable file "stp" about two thirds of the way down figure 4, on the right. The expression "-geometry +0+120" is an X instruction that tells StP that the starting x,y pixel position for StP is 0,120. This is needed to insure that the StP windows do not overlay the ETI top-level window (figure 1). The user may arbitrarily select any x pixel position that lies on the screen. The y pixel position is critical in that it should be chosen so that the tool's windows are positioned below the ETI top-level window.

Now, further down in figure 4, find the "Emacs" editor. The expression "xterm -fn 10×20 -e emacs" indicates that (1) the font size for the emacs editor is 10×20 pixel fixed-width font (the emacs font size was increased from the default size to increase readability), (2) emacs is executed inside the xterm window.

4.3.2 An Example of Tailoring to a Specific Project

Figure 5 shows the top-level window for a hypothetical project (when all tools and options are displayed). This project does not use Ada but uses C and FORTRAN. Note how this top-level window differs from the one given in figure 2. The label at the top of the screen is now "PROJECT X". The category, "FORTRAN Toolset", was created (fourth category). For Project X, a number of project-specific tools are used, so the next tool category, "Project X Tools", was created. Finally, two different CASE tools are used, Code Center and Cadre Teamwork.

Figure 6 shows the changes that were made to the setup table to tailor the ETI to a specific project. The changes were made to the file "sweep_config". After making the appropriate changes, the user types

```
sweep
```

to bring up the ETI as shown in figure 5.

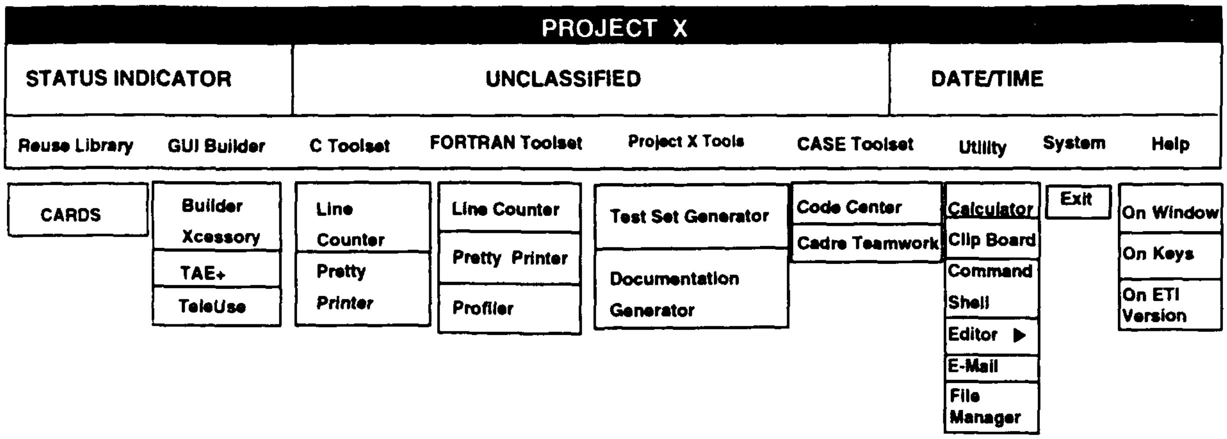
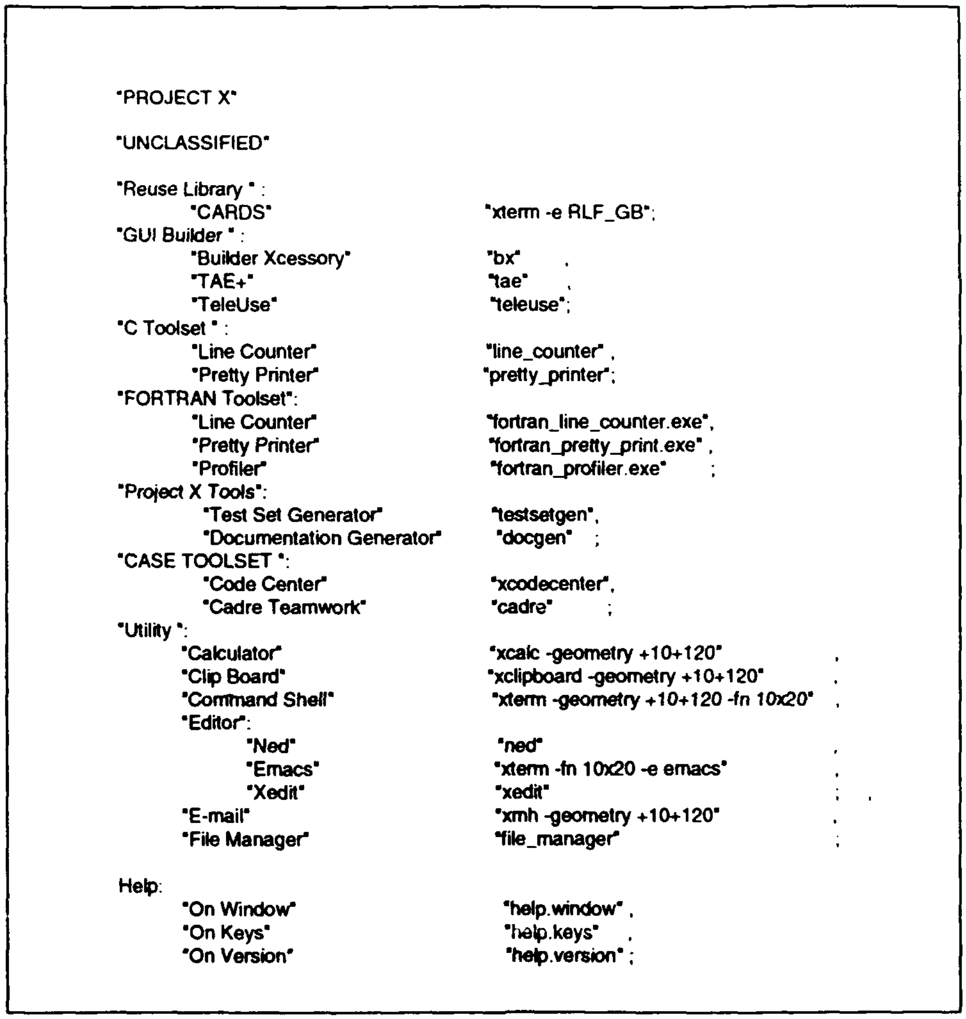


Figure 5. Top-level window with tools and options displayed tailored to a specific project.



```
"PROJECT X"

"UNCLASSIFIED"

"Reuse Library " :
        "CARDS"                     "xterm -e RLF_GB";
"GUI Builder " :
        "Builder Xcessory"          "bx"      ,
        "TAE+"                      "tae"     ,
        "TeleUse"                   "teleuse";
"C Toolset " :
        "Line Counter"              "line_counter" ,
        "Pretty Printer"            "pretty_printer";
"FORTRAN Toolset":
        "Line Counter"              "fortran_line_counter.exe",
        "Pretty Printer"            "fortran_pretty_print.exe" ,
        "Profiler"                  "fortran_profiler.exe"     ;
"Project X Tools":
         "Test Set Generator"        "testsetgen",
         "Documentation Generator"   "docgen"   ;
"CASE TOOLSET ":
         "Code Center"              "xcodecenter",
         "Cadre Teamwork"           "cadre"        ;
"Utility ":
       "Calculator"                 "xcalc -geometry +10+120"              ,
       "Clip Board"                 "xclipboard -geometry +10+120"         ,
       "Command Shell"              "xterm -geometry +10+120 -fn 10x20"    ,
       "Editor":
               "Ned"                 "ned"                                 ,
               "Emacs"               "xterm -fn 10x20 -e emacs"            ,
               "Xedit"               "xedit"                               ;  ,
       "E-mail"                      "xmh -geometry +10+120"               ,
       "File Manager"                "file_manager"                        ;

Help:
       "On Window"                    "help.window" ,
       "On Keys"                      "help.keys"   ,
       "On Version"                   "help.version" ;
```

Figure 6. Example of Setup Table Tailored to a specific project.

5.0 LESSONS LEARNED

This section describes the lessons that were learned in a number of areas during the development of the ETI. Lessons learned include the use of Ada versus C, the choice of the GUI specification, tailoring approaches, and sources for public domain tools.

Lessons have been learned in a number of areas during the development of the ETI.

5.1 VALUE OF DEMONSTRATIONS

Task members were successful in finding a transition project and users for the ETI by rapidly developing a functional prototype which was demonstrated repeatedly. Task members felt that the initial prototype offered enough functionality that project developers would recognize its potential. Demonstrations were given to key personnel of candidate projects at NRaD and other Navy laboratories. Private industry users were found by giving demonstrations at national AFCEA conferences.

Demonstrations led to ETI improvements as a result of user feedback. Examples of helpful suggestions include (1) making the entire top-level window tailorable, (2) adding a tool search capability for finding the category where a tool is located, and (3) including a context-sensitive help capability. The final capability allows a user to get help on a particular field by positioning the cursor on the field and then clicking the mouse button.

5.2 ADVANTAGE OF FINDING USERS EARLY

Early in the development of the ETI the NRaD OSS project members made many valuable suggestions including providing the capability for subcategories of tools and reducing the size of the top-level window so that other application windows were visible.

5.3 ADA VERSUS C FOR DEVELOPMENT

The ETI was developed using the C GUI builder, Builder Xcessory. When the ETI development started, the only Ada GUI builder available was TAE+, and it lacked maturity.

If we were to start development today, we would probably do it in Ada. TAE+ has improved, and a number of other Ada compatible GUI builders have appeared in the market place.

5.4 OPEN LOOK VERSUS MOTIF

The first prototype of the ETI, which was written using Open Look, was demonstrated to managers and technical representatives of NRaD projects. Acceptance was low because these projects had chosen Motif as the user interface standard. Once we changed to Motif, customer enthusiasm increased significantly.

5.5 TAILORING

Three tailoring approaches were considered. The first approach was to have the user employ a GUI builder to customize the ETI to meet his or her needs. The second was to provide an automated method for constructing the windowing framework from a setup table. The third was to extend the second approach by building the setup table from an interactive program. The second approach was followed. The first approach was not chosen because it requires the user to know Motif and how to use a GUI builder. The third was not done with the ETI, version 1.1, because of time constraints and because it would probably not be significantly easier for the user.

5.6 SOURCES FOR PUBLIC DOMAIN TOOLS

The source of most Ada public domain tools was the Ada Software Repository (ASR) at White Sands, New Mexico, and the AdaNet repository in Dellslow, West Virginia. Sources for C tools were found in the Archie bulletin board (Ref. 4). The utility programs are from the X11R4 libraries. Considerable time and effort was spent compiling and executing tools from the Ada repositories. Some tools worked, some did not work well, and some did not work at all. "House cleaning" at repositories needs to be done with the help of the user community.

Lessons learned searching for `"good"` public domain tools will be briefly discussed. The ASR and AdaNet each contained six Ada pretty printers. Only Pretty Printer 4 and pretty printer 6 were written in valid Ada. Pretty Printer 6 was selected because task members could not get Pretty Printer 4 to execute properly. Note that Pretty Printer 6 contains some minor bugs.

Two Ada body stubbers were found in the ASR and AdaNet. Neither execute when a validated Ada compiler is used. Body Stubber 2 was modified by task members so that it does run on validated Ada compilers. This modified body stubber is being incorporated into the December 1992 release of the Ada Language System/Navy (ALS/N).

Several Ada line counters were examined in the two repositories. The line counter `File_Checker` was selected. Several bugs were found in it and fixed.

The C line counter selected, kdsi, was located in the Archie bulletin board (Ref. 4). One minor bug (for an uninitialized variable) was fixed before it was integrated into the ETI. Archie can be accessed by rlogin-ing or telnet-ing to `archie.ans.net` (147.225.1.2) or `archie.sura.net` (128.167.254.179) with username `"archie"` and no password.

The C pretty printer currently used is from SunOS 4.1.1. For portability reasons, the next version of the ETI will probably use the C pretty printer, cpr, which is listed in the Archie bulletin board (Ref. 4).

5.7 HUMAN FACTORS INVOLVEMENT

A human factors expert, Dr. K. Fernandes of NRaD, periodically evaluated the ETI. She provided invaluable comments regarding simplifying the user interface. Her user interface standard was followed (Ref. 1). Her comments related to consistency, readability, and other aspects covered in the interface standard.

6.0 RECOMMENDATIONS

6.1 RECOMMENDATIONS FOR FUTURE ETI RESEARCH

Below is a list of recommendations for future ETI research and development:

1. Make the following improvements to the ETI:

   - Add tool search capability to find category where tool is located.
   - Add comprehensive error messages for setup table.
   - Add context-sensitive help capability.

2. Port the ETI to additional Unix-based computers.

3. Write Motif front-ends to the Ada tools. This front-end is especially needed for the pretty printer.

4. Include additional Ada tools. Those being considered for inclusion are Halstead metrics, McCabe metrics, path analyzer, performance analyzer, and statement profiler from AdaNet. Others that have wide application are also candidates.

5. Include C++ tools and write Motif front-ends for these tools. These include a pretty printer, a line counter, and others that are in the public domain and have wide application.

6. Include additional C tools and write Motif front-ends for them. These tools include a cross referencer, prototypes declaration generator, and others.

7. Write Motif front-ends for commercial CASE tools to facilitate their use. These front-ends facilitate the use of startup scripts to define tool parameters. Tools may include CADRE Teamwork and Interactive Development Environments' (IDE) Software Thru Pictures.

6.2 RECOMMENDATIONS FOR SOFTWARE DEVELOPERS

The following are general recommendations based on what we observed while developing the ETI:

1. Developers of DoD software and CASE tool developers should have human factors specialists evaluate early prototypes of their user interfaces. This will help to eliminate awkwardness and inconsistencies, and to simplify the use of the tools.

2. Software developers should use standards to help them produce user interfaces that are consistent in appearance and in operation.

7.0 REFERENCES

1. Fernandes, K. 1992. "User Interface Specifications for Command and Control Systems," Version 1.0, Naval Command, Control and Ocean Surveillance Center, Research, Development, Test and Evaluation Division, San Diego, CA.

2. Naval Command, Control and Ocean Surveillance Center, Research, Development, Test and Evaluation Division. 1992. "CRSS Replicated Site Procedures Manual, Alpha Release 1.0," San Diego, CA.

3. Converse, D., J. Fulton, M. Leger, K. Packard, C. Peterson, R. Scheifler, and R. Swick. "User Commands for X Window System," on-line instructions, Massachusetts Institute of Technology Consortium, X Version 11, Release 4, Boston, MA.

4. Deutsch, P., A. Emtage, B. Heelen, and M. Parker. 1992. "Archie Database," Archie Group, McGill University, Montreal, Canada.

5. Sun Microsystems, Inc. 1990. "SunOS Reference Manual, Revision A."

6. "Ada Style Guide Handbook." 1988. MIL-HDBK-1804, Draft, National Aeronautics and Space Administration/Goddard Space Flight Center

8.0 BIBLIOGRAPHY

MountainNet. 1991. "The AdaNet Repository (ASV2) Catalog of Online Holdings," Dellslow, WV.

Tran, N., and H. Mumm, July, 1992. "User's Instructions for the SWEEP Environment/ Tool Integrator," Version 1.0, Draft, Naval Command, Control and Ocean Surveillance Center, Research, Development, Test and Evaluation Division (NRaD), San Diego, CA.

Open Software Foundation. *OSF/Motif Style Guide*, Revision 1.1 (For OSF/Motif Release 1.1), Prentice Hall, Englewood Cliffs, NJ.

Quercia, V., and T. O'Reilly. 1989. *X Window System User's Guide*, Vol. 3, O'Reilly and Associates, Inc., Sebastapol, CA.

Appendix A

USING THE TOOLS

This appendix briefly describes the tools that are included with the ETI and shows in detail how to use them. Specific examples are provided. The test files that are described in the examples below are included with version 1.1 of the ETI. Users may run the examples exactly as given here.

A.1 C TOOLSET

Line Counter

Function: The line counter is the "kdsi" program written by B. Renaud. It was obtained from Ohio State University. A Motif front-end was added. Line counter reads in text files and displays the number of lines of code, blank lines, comment lines, and number of comments to the user's monitor.

Example: The user provides input through the menu given in figure A-1. In this example, the user wanted the counts for all C files in the directory "$SWEEP_HOME/ testfiles". This was indicated in the "Input" box. The user executed the tool by clicking on the "Execute" box. The bottom panel displays the lines of code, blank lines, comment lines, and number of comments for all C files in the designated directory.

To output counts to both the user's monitor and a file, the user must enter a file name in the "Output" box.

The "Display available files" box allows a user to list files.

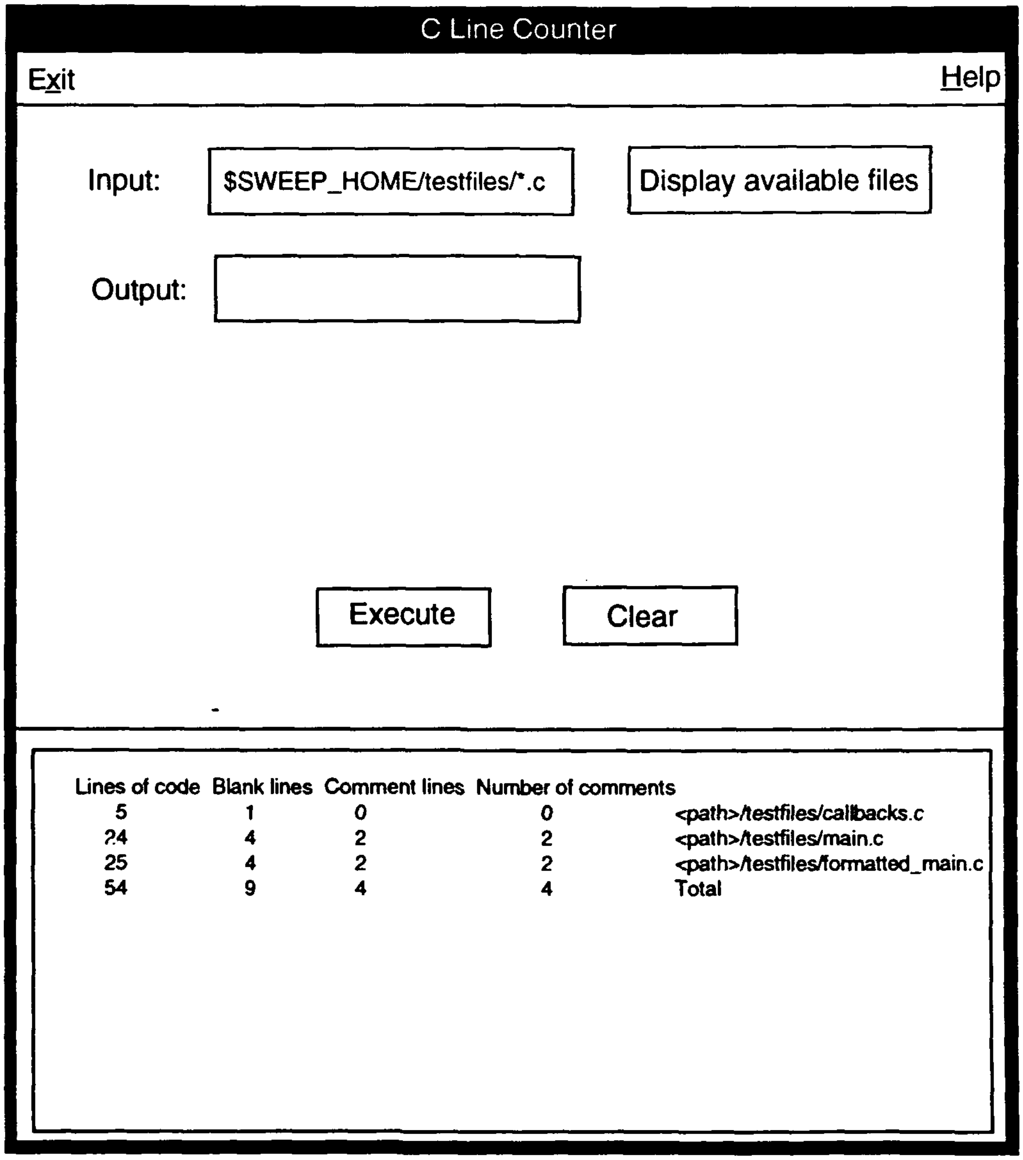
Add:tional information on the line counter may be found by examining the source code and "readme" files that are in the directory "$SWEEP_HOME/tools/c/ kdsi" on the user's computer.

Pretty Printer

Function: This pretty printer is the C beautifier (cb) program in Ref. 5. A Motif front-end was added. The pretty printer reads in a C source code file and produces a source file with indentation. The user can join split lines and can split lines that are longer than a user-specified length.

Example: The user provides input through the menu given in figure A-2. In this example the user wanted the file "$SWEEP_HOME/testfiles/main.c" to be reformatted into standard C style and to be output to the file "<path>/sweep1.1/testfiles/reformatted_main.c". The user must provide the remainder of the path name, which is indicated by <path>. To run the pretty printer, the user entered these file names in the appropriate boxes and clicked on the option for standard C style. The output was both displayed in the bottom panel of figure 8 and written to the designated file.

To direct the output to the screen only, the user does not provide an output file name. Options are also available to join split lines and to split lines. These options may be invoked simultaneously.

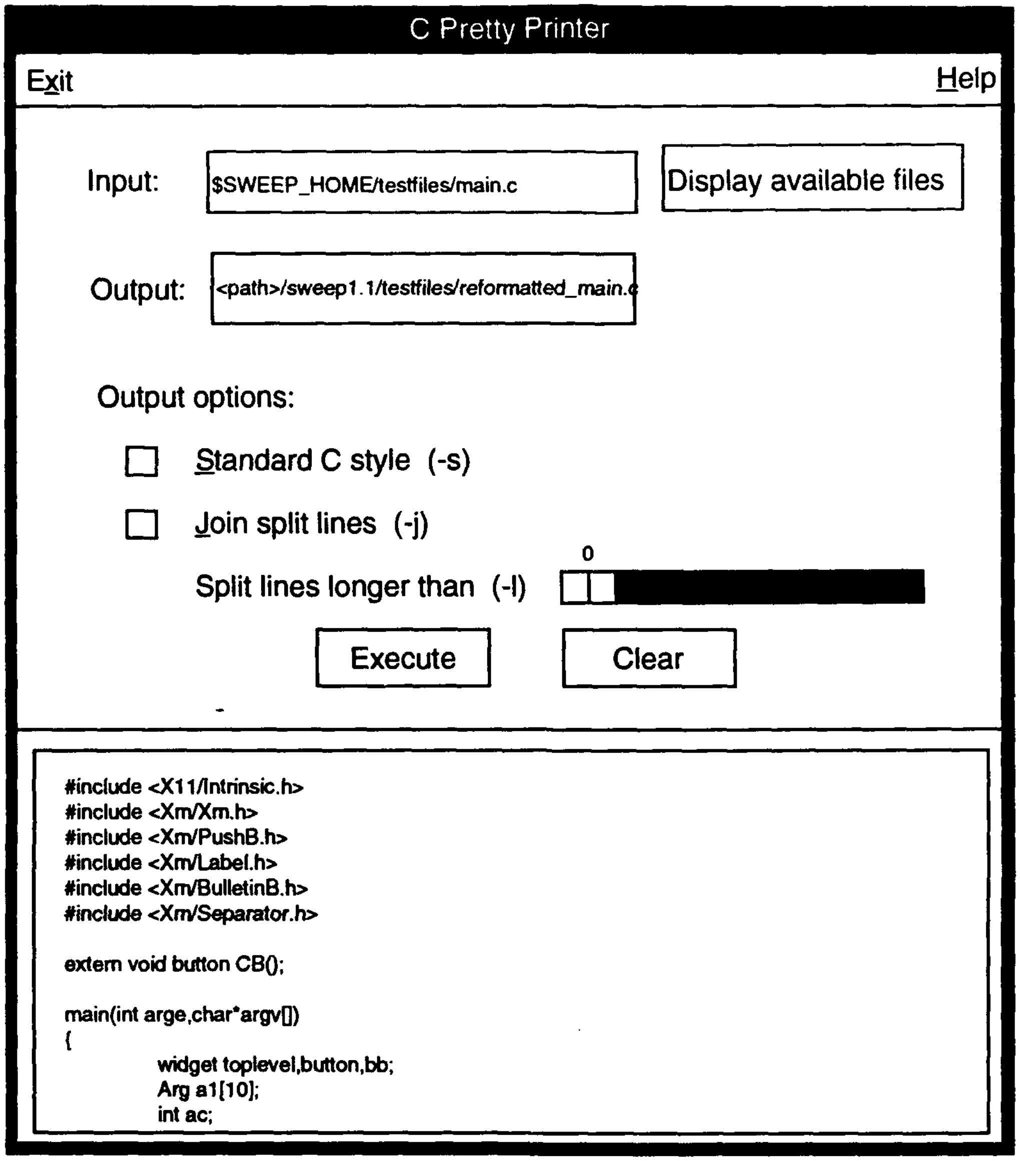


Input: $SWEEP_HOME/testfiles/*.c

Output:

```
Lines of code  Blank lines  Comment lines  Number of comments
     5              1             0                 0            <path>/testfiles/callbacks.c
    24              4             2                 2            <path>/testfiles/main.c
    25              4             2                 2            <path>/testfiles/formatted_main.c
    54              9             4                 4            Total
```

Figure A-1. Menu for C line counter.



Input: $SWEEP_HOME/testfiles/main.c

Output: <path>/sweep1.1/testfiles/reformatted_main.

Output options:

□ Standard C style (-s)

□ Join split lines (-j)

Split lines longer than (-l) 0

```
#include <X11/Intrinsic.h>
#include <Xm/Xm.h>
#include <Xm/PushB.h>
#include <Xm/Label.h>
#include <Xm/BulletinB.h>
#include <Xm/Separator.h>

extern void button CB();

main(int arge,char*argv[])
{
        widget toplevel,button,bb;
        Arg a1[10];
        int ac;
```

Figure A-2. Menu for C pretty printer.

Within the ETI, the user may examine the contents of the output file by going to UTILITY and Command Shell and then typing

```
more <path>/sweep1.1/testfiles/reformatted_main.c
```

Additional information on the pretty printer is found in Ref. 5, page 53.

A.2 ADA TOOLSET

Body Stubber

Function: Body stubber creates an Ada package body with stubs for subprograms and tasks from an Ada package specification that contains the specifications for subprograms and tasks.

Input: Ada_Package_Spec -- Ada package specification with subprogram and task specifications

Output: Ada_Package_Body -- Ada package body with stubs for subprograms and tasks

Example: Below is what a typical session looks like immediately after the body stubber is invoked:

Enter name of file:
`$SWEEP_HOME/testfiles/stubber_input`

Enter output file name(Default:`$SWEEP_HOME/testfiles/stubber_input_body`):
RETURN -- When user presses return default file is created.

Compilation Order Maker

The Compilation Order Maker, being developed by NRaD, determines the proper compilation order of a collection of files.

Line Counter

Function: Counts Ada source code statements, comments, and lines (card-image statements).

Input: Ada_Source_File -- File containing Ada source code

or

@Ada_Source_Files -- File containing names of source code

Output: Lines_of_Code_File -- File containing line counts.

Example: Below is an example of a session when the line counter is invoked:

FILE CHECKER, Version 1.4

```
Name of Output File (RETURN to Abort)?
$SWEEP_HOME/testfiles/pretty_print_output
                             -- User provides output file name.
File Name > $SWEEP_HOME/testfiles/pretty_print_input
                             -- This test file is provided.
File Name > RETURN           -- Press RETURN to terminate program.
```

Pager

Function: Pager combines many files into a concatenated file and breaks a concatenated file into multiple files. The information below is for breaking a concatenated file into multiple files.

Input: Large_Ada_Source_File -- This is concatenated file in pager format

Output: Multiple_Files -- Individual files

Example:

The example given below breaks a concatenated file into individual files. The user can see other pager options that are available by entering "h" for help:

```
PAGER2, Modified Pager2, Ada Version 1.1
type 'h' for Help
PAGER2> u $SWEEP_HOME/testfiles/pager_input

Extracting compilation.txt -- 5 lines
Extracting cas3.a -- 301 Lines
Extracting list.a -- 304 Lines
Extracting fcheck.a -- 327 Lines
PAGER2> x                   -- Exits user from pager.
```

Pretty Printer

Function: Reads in Ada source code files and creates an output file with spacing, indentation, and capitalization that conforms to the proposed Ada Style Guide Handbook (Ref.6).

Input: Ada_Source_File -- File containing Ada source code,

or

@Ada_Source_Files -- File containing Ada source code file names.

Output: Reformatted_Ada_Source -- Reformatted Ada source code

Example:

Below is an example of a session after the line counter is invoked.

```
Ada File? > $SWEEP_HOME/testfiles/pretty_print_input

$SWEEP_HOME/testfiles/pretty_print_inppp created
                 -- File of reformatted Ada source code.

$SWEEP_HOME/testfiles/pretty_print_inpsr created
                 -- File containing statistical
                 -- information on source code.
```

A.3 UTILITY

A.3.1 Calculator, Clip Board, Command Shell, and E-Mail

The calculator, clip board, command shell, and E-Mail tools are provided by the X Window System. User information on these tools can be found on-line by using `"man pages"`. For example, type `"man xcalc"`, `"man xclipboard"`, `"man xterm"`, `"man xmh"` at the Unix prompt to display manual pages for the calculator, clip board, command shell, and e-mail, respectively.

A.3.2 Editors

A.3.2.1 NED. The Navy Command and Control System Ashore Editor (NED) is a text display and editing system that was designed for viewing portions of source code and documentation stored in the CRSS library. NED may also run as a stand-alone application for editing text files. NED was developed by K. Allen, Intermetrics for the OSS project. NED user instructions are contained in Ref. 2.

A.3.2.2 Emacs. Emacs is the GNU Emacs editor from the Free Software Foundation. The editor includes facilities to send and receive mail, run subprocesses, and do compilations. It was written by R. Stallman.

A.3.2.3 Xedit. Xedit is a customizable text editor for the X Window System developed by MIT. It was written by C. Peterson, MIT X Consortium.

A.3.3 File Manager

File Manager is a Motif application used to navigate through file hierarchies. This tool provides facilities to search, move, copy, compress, and view files, and to obtain file information. It was developed by K. Allen of Intermetrics for the OSS project.

Appendix B

BACKGROUND INFORMATION ON THE ADA TOOLS

Ada Primitive Compilation Order Tool (APRICOT)

APRICOT is a tool for determining the compilation order of a collection of Ada files. It is being developed by R. Ollerton of NRaD. It currently runs on VAX/VMS and will be ported to Suns and additional platforms. It also performs the functions of pager and creates command compilation files from concatenated pager files.

Body Stubber

The body stubber is Body Stubber 2 from the ASR, White Sands, New Mexico. It was written by J. Orost of Concurrent Computer Corporation and was upgraded by N. Tran of NRaD to run on VAX/VMS (DEC Ada), Sun (Verdix), and other validated Ada compilers. This tool is especially useful for large projects, where interfaces must be defined very early. The body stubber reads in an Ada package specification that contains the specification for subprograms and tasks. It automatically creates a compilable package body containing stubs for the subprograms and tasks. It was resubmitted to the AdaNet Repository in December 1991 by NRaD.

Line Counter

The Ada line counter is File_Checker from the ASR. It was written by R. Conn of Texas Instruments (TI) and the Management Assistance Corporation of America (MACA). Fixes have been made by R. Mumm of NRaD and P. Babick of Science Applications International Corporation (SAIC). The line counter is written in Ada and runs on VAX/VMS (DEC Ada), Sun (Verdix), and other validated Ada compilers. The tool counts Ada source lines of code several ways. It was used on the Ada Bit-Oriented Message Handler (ABOM) project by NRaD and SAIC to report programmer productivity statistics. It was resubmitted to the AdaNet in December 1991.

Pager

The pager is Pager2 from the ASR. It was developed by Richard Conn, Texas Instruments and MACA. It was written in Ada and runs on VAX/VMS (DEC Ada), Verdix (Sun), IntegrAda, and other validated Ada compilers. The tool concatenates Ada source files, breaks concatenated files into individual files, and makes listings. Pager requires that source files be in the pager format, a prefixed banner of comment statements. Pager is a tool that is used for storing and transporting related files. It is used by ASR and AdaNet users to separate concatenated files.

Pretty Printer

This pretty printer is Pretty Printer 6 from the ASR. It was written by A. Shell of AdaCraft, Inc., for the NASA/Goddard Space Flight Center. It was written in Ada and runs on VAX/VMS (DEC Ada), SUN (Verdix), PC (Alsys), and other compilers/computers. Pretty Printer 6 implements many of the directives in Ref. 6. The modifications required to change the number of columns of indentation, the case of variable names, and other pretty printer parameters are isolated in one Ada package specification. This pretty printer was used by NRaD for the Ada discrete-event simulation research conducted by the Shared Adaptive Internetworking (SAINT) project.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
August 1992

3. REPORT TYPE AND DATES COVERED
Final: 10/91 — 9/92

4. TITLE AND SUBTITLE
ENVIRONMENT/TOOL INTEGRATOR FOR SOFTWARE DEVELOPMENT
Version 1.1

5. FUNDING NUMBERS
PROG: 0602234N
PROJ: ECB3
ACCESS: DN088690

6. AUTHOR(S)
N. T. Tran and R. H. Mumm

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Command, Control and Ocean Surveillance Center (NCCOSC)
RDT&E Division (NRaD)
San Diego, CA 92152-5000

8. PERFORMING ORGANIZATION REPORT NUMBER
NRaD TR 1548

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Technology
Ballston Tower
800 North Quincy
Arlington, VA 22267-5660

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report describes the environment/tool integrator (ETI) that was developed under the Software Engineering Environment Prototypes (SWEEP) task of the Software Engineering for Command Control and Communications (C^3) Systems project. The ETI is a windowing framework for accessing the diverse collections of software tools used by software development projects. It can easily be customized to satisfy the unique needs of specific projects.

14. SUBJECT TERMS
C, Ada
Motif, X
environment/tool integrator
graphical user interface builder

15. NUMBER OF PAGES
38

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
UNCLASSIFIED

18. SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

19. SECURITY CLASSIFICATION OF ABSTRACT
UNCLASSIFIED

20. LIMITATION OF ABSTRACT
SAME AS REPORT

UNCLASSIFIED

21a. NAME OF RESPONSIBLE INDIVIDUAL N. Tran	21b. TELEPHONE *(Include Area Code)* (619) 553–4076	21c. OFFICE SYMBOL Code 411

INITIAL DISTRIBUTION

Code 0012	Patent Counsel	(1)
Code 01	R. T. Shearer	(1)
Code 014	W. T. Rasmussen	(1)
Code 0142	K. J. Campbell	(1)
Code 144	V. Ware	(1)
Code 40	R. C. Kolb	(1)
Code 402	R. A. Wasilausky	(1)
Code 41	A. G. Justice	(1)
Code 411	J. Schulte	(1)
Code 411	H. Mumm	(20)
Code 834	B. Barlin	(1)
Code 961	Archive/Stock	(6)
Code 964B	Library	(2)

Defense Technical Information Center
Alexandria, VA 22304-6145 (4)

NCCOSC Washington Liaison Office
Washington, DC 20363-5100

Center for Naval Analyses
Alexandria, VA 22302-0268

Navy Acquisition, Research & Development
Information Center (NARDIC)
Washington, DC 20360-5000

GIDEP Operations Center
Corona, CA 91718-8000

National Security Agency
Fort Meade, MD 20755-6000

Office of Under Secretary of Defense
Arlington, VA 20301-3080

Defense Advanced Research
Projects Agency
Arlington, VA 22203-1714 (3)

Chief of Naval Operations
Washington, DC 20350-2000

Office of Naval Technology
Arlington, VA 22217-5000 (4)

Office of Naval Research
Arlington, VA 22217-5000 (2)

Space and Naval Warfare
Systems Command
Washington, DC 20363-5100 (3)

Naval Research Laboratory
Washington, DC 20375-5000 (4)

Naval Sea Systems Command
Washington, DC 20362-5101 (3)

Naval Surface Warfare Center
Dahlgren, VA 22448-5000

Naval Surface Warfare Center
Silver Spring, MD 20903-5000 (4)

Naval Undersea Warfare Center
Newport, RI 02841-5047 (3)

Naval Undersea Warfare Center
Detachment
New London, CT 06320

ADA Joint Program Office
Washington, DC 20301-3081

Naval Air Warfare Center
Weapons Division
China Lake, CA 93555-6001

Washington Navy Yard
Washington, DC 20374 (2)

Naval Postgraduate School
Monterey, CA 93943

Rome Laboratory
Griffiss AFB, NY 13441-5700 (2)

U. S. Army, Headquarters
Washington, DC 20301

U. S. Army Ballistic Research Laboratory
Aberdeen, MD 21005-5066

Carnegie Mellon University
Pittsburgh, PA 15213-3890

Georgia Institute of Technology
Atlanta, GA 30332

Asset Source for Software Engineering
Technology
Morgantown, WV 26505

Charles Stark Draper Laboratory, Inc.
Cambridge, MA 02139-3563

Mitre Corporation
McLean, VA 22102

Northrop Corporation
Hawthorne, CA 92050

Paramax
Paoli, PA 19301

VISICOM
San Diego, CA 92127